INTERNATIONAL STANDARD

ISO/IEC 15693-3

Second edition
2009-04-15

Identification cards — Contactless integrated circuit cards — Vicinity cards —

Part 3: Anticollision and transmission protocol

Cartes d'identification — Cartes à circuit(s) intégré(s) sans contact — Cartes de voisinage —

Partie 3: Anticollision et protocole de transmission

Reference number
ISO/IEC 15693-3:2009(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 15693-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

This second edition cancels and replaces the first edition (ISO/IEC 15693-3:2001), Table 1 and 9.4.2 of which have been technically revised and Figure 10 redrawn for clarity.

ISO/IEC 15693 consists of the following parts, under the general title *Identification cards — Contactless integrated circuit cards — Vicinity cards*:

— *Part 1: Physical characteristics*

— *Part 2: Air interface and initialization*

— *Part 3: Anticollision and transmission protocol*

Introduction

ISO/IEC 15693 is one of a series of International Standards describing the parameters for identification cards as defined in ISO/IEC 7810 and the use of such cards for international interchange.

This part of ISO/IEC 15693 describes the anticollision and transmission protocols.

This part of ISO/IEC 15693 does not preclude the incorporation of other standard technologies on the card.

Contactless card standards cover a variety of types as embodied in ISO/IEC 10536 (close-coupled cards), ISO/IEC 14443 (proximity cards) and ISO/IEC 15693 (vicinity cards). These are intended for operation when very near, nearby and at a longer distance from associated coupling devices, respectively.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from:

JP 2561051 - Circuit Structure of Inductive Contactless Responding Unit JP 2981517, JP 2129209 – Read to Verify Written Data	OMRON Corporation Intellectual Property Group 20 Igadera, Shimokaiinji Nagaokakyo-City Kyoto 617-8510 Japan
US5793324 EP831618 EP837412 EP845751	Texas Instruments Deutschland GMBH TIRIS Haggarty Strasse 1 8050 Freising Germany

The subject matter of these patents is anticollision, affecting Clause 8 of this part of ISO/IEC 15693.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Identification cards — Contactless integrated circuit cards — Vicinity cards

Part 3: Anticollision and transmission protocol

1 Scope

This part of ISO/IEC 15693 specifies:

— protocol and commands,

— other parameters required to initialize communications between a vicinity integrated circuit card and a vicinity coupling device,

— methods to detect and communicate with one card among several cards ("anticollision"),

— optional means to ease and speed up the selection of one among several cards based on application criteria.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7816-6:2004, *Identification cards — Integrated circuit cards — Part 6: Interindustry data elements for interchange*

ISO/IEC 13239, *Information technology — Telecommunications and information exchange between systems — High-level data link control (HDLC) procedures*

ISO/IEC 15693-1, *Identification cards — Contactless integrated circuit(s) cards — Vicinity cards — Part 1: Physical characteristics*

ISO/IEC 15693-2, *Identification cards — Contactless integrated circuit cards — Vicinity cards — Part 2: Air interface and initialization*

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15693-1, ISO/IEC 15693-2 and the following apply.

3.1.1
anticollision loop
algorithm used to prepare for and handle a dialogue between a VCD and one or more VICCs from several in its energizing field

3.1.2
byte
string that consists of 8 bits of data designated b1 to b8, from the most significant bit (MSB, b8) to the least significant bit (LSB, b1)

3.2 Abbreviated terms

AFI	application family identifier
CRC	cyclic redundancy check
DSFID	data storage format identifier
EOF	end of frame
LSB	least significant bit
LSByte	least significant byte
MSB	most significant bit
MSByte	most significant byte
RFU	reserved for future use
SOF	start of frame
UID	unique identifier
VCD	vicinity coupling device
VICC	vicinity integrated circuit card

3.3 Symbols

f_c	frequency of operating field (carrier frequency)

4 Definition of data elements

4.1 Unique identifier (UID)

The VICCs are uniquely identified by a 64 bits unique identifier (UID). This is used for addressing each VICC uniquely and individually, during the anticollision loop and for one-to-one exchange between a VCD and a VICC.

The UID shall be set permanently by the IC manufacturer in accordance with figure 1.

MSB LSB

64 57	56 49	48 1
'E0'	IC Mfg code	IC manufacturer serial number

Figure 1 — UID format

The UID comprises

- The MSByte (bits 64 – 57) shall be 'E0',
- The IC manufacturer code (bits 56 – 49) according to ISO/IEC 7816-6:2004,
- A unique serial number (bits 48 – 1) assigned by the IC manufacturer.

4.2 Application family identifier (AFI)

AFI (Application family identifier) represents the type of application targeted by the VCD and is used to extract from all the VICCs present only the VICCs meeting the required application criteria.

It may be programmed and locked by the respective commands.

AFI is coded on one byte, which constitutes 2 nibbles of 4 bits each.

The most significant nibble of AFI is used to code one specific or all application families, as defined in table 1.

The least significant nibble of AFI is used to code one specific or all application sub-families. Sub-family codes different from 0 are proprietary.

Table 1 — AFI coding

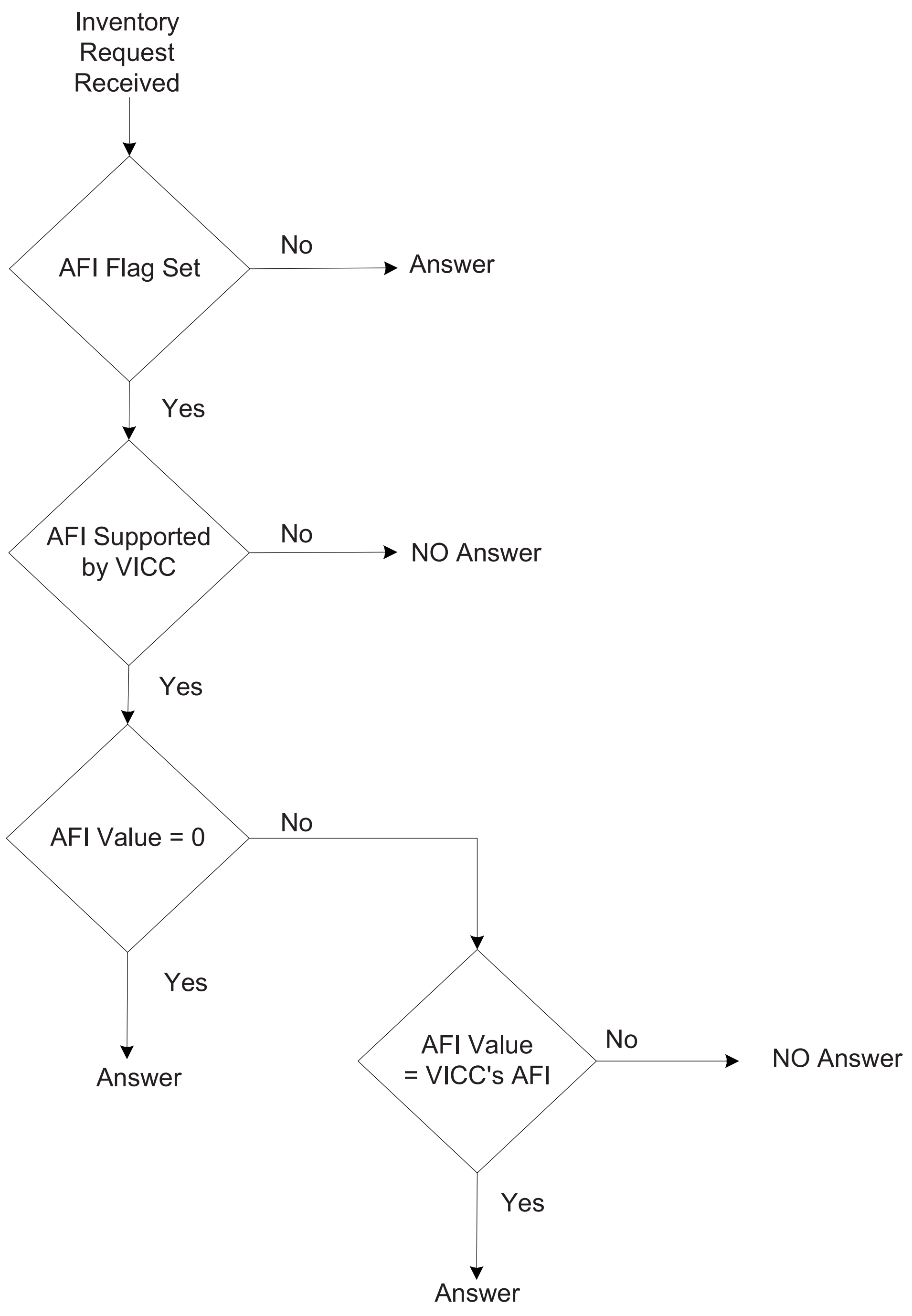
AFI most significant nibble	AFI least significant nibble	Meaning VICCs respond from	Examples / note
‘0’	‘0’	All families and subfamilies	No applicative preselection
X	'0'	All sub-families of family X	Wide applicative preselection
X	Y	Only the Yth sub-family of family X	
‘0’	Y	Proprietary sub-family Y only	
‘1'	‘0’, Y	Transport	Mass transit, Bus, Airline
'2'	‘0’, Y	Financial	IEP, Banking, Retail
'3'	‘0’, Y	Identification	Access control
'4'	‘0’, Y	Telecommunication	Public telephony, GSM
‘5’	‘0’, Y	Medical	
'6'	‘0’, Y	Multimedia	Internet services
'7'	‘0’, Y	Gaming	
'8'	‘0’, Y	Data storage	Portable files
'9'	‘0’, Y	EAN-UCC system for Application Identifiers	Managed by ISO/IEC JTC 1/SC 31
'A'	‘0’, Y	Data Identifiers as defined in ISO/IEC 15418	Managed by ISO/IEC JTC 1/SC 31
'B'	‘0’, Y	UPU	Managed by ISO/IEC JTC 1/SC 31
'C'	‘0’, Y	IATA	Managed by ISO/IEC JTC 1
'D'	‘0’, Y	RFU	Managed by ISO/IEC JTC 1/SC 17
'E'	‘0’, Y	RFU	Managed by ISO/IEC JTC 1/SC 17
‘F’	‘0’, Y	RFU	Managed by ISO/IEC JTC 1/SC 17

NOTE X = ‘1’ to ‘F’, Y = ‘1’ to ‘F’.

The support of AFI by the VICC is optional.

If AFI is not supported by the VICC and if the AFI flag is set, the VICC shall not answer whatever the AFI value is in the request.

If AFI is supported by the VICC, it shall answer according to the matching rules described in table 1.



NOTE "Answer" means that the VICC shall answer to the Inventory request.

Figure 2 — VICC decision tree for AFI

4.3 Data storage format identifier (DSFID)

The Data storage format identifier indicates how the data is structured in the VICC memory.

It may be programmed and locked by the respective commands. It is coded on one byte. It allows for instant knowledge on the logical organisation of the data.

If its programming is not supported by the VICC, the VICC shall respond with the value zero ('00').

4.4 CRC

The CRC shall be calculated in accordance with ISO/IEC 13239.

The initial register content shall be all ones: 'FFFF'.

The two bytes CRC are appended to each request and each response, within each frame, before the EOF. The CRC is calculated on all the bytes after the SOF up to but not including the CRC field.

Upon reception of a request from the VCD, the VICC shall verify that the CRC value is valid. If it is invalid, it shall discard the frame and shall not answer (modulate).

Upon reception of a response from the VICC, it is recommended that the VCD verifies that the CRC value is valid. If it is invalid, actions to be performed are left to the responsibility of the VCD designer.

The CRC is transmitted least significant byte first.

Each byte is transmitted least significant bit first.

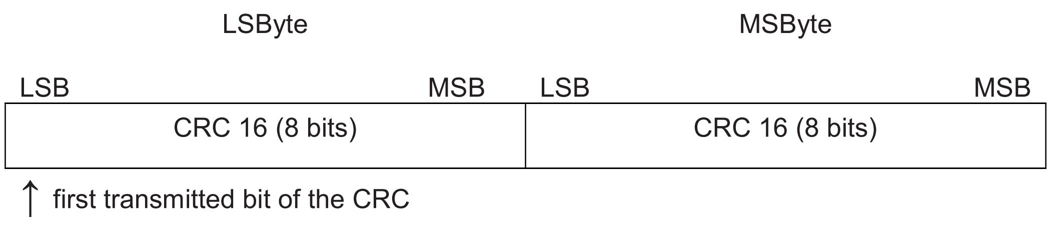


Figure 3 — CRC bits and bytes transmission rules

5 VICC memory organization

The commands specified in this part of ISO/IEC 15693 assume that the physical memory is organized in blocks (or pages) of fixed size.

— Up to 65536 blocks can be addressed.

— Block size can be of up to 256 bits.

— This leads to a maximum memory capacity of up to 2 MBytes (16 MBits).

The commands described in this part of ISO/IEC 15693 allow the access (read and write) by block(s). There is no implicit or explicit restriction regarding other access method (e.g. by byte or by logical object in future revision(s) of this part of ISO/IEC 15693 or in custom commands).

6 Block security status

The block security status is sent back by the VICC as a parameter in the response to a VCD request as specified in clause 10 (e.g. Read single block). It is coded on one byte.

It is an element of the protocol. There is no implicit or explicit assumption that the 8 bits are actually implemented in the physical memory structure of the VICC.

Table 2 — Block security status

Bit	Flag name	Value	Description
b1	Lock_flag	0	Not locked
		1	Locked
b2 to b8	RFU	0	

7 Overall protocol description

7.1 Protocol concept

The transmission protocol (or protocol) defines the mechanism to exchange instructions and data between the VCD and the VICC, in both directions.

It is based on the concept of "VCD talks first".

This means that any VICC shall not start transmitting (i.e. modulating according to ISO/IEC 15693-2) unless it has received and properly decoded an instruction sent by the VCD.

a) the protocol is based on an exchange of

— a request from the VCD to the VICC

— a response from the VICC(s) to the VCD

The conditions under which the VICC sends a response are defined in clause 10.

b) each request and each response are contained in a frame. The frame delimiters (SOF, EOF) are specified in ISO/IEC 15693-2.

c) each request consists of the following fields:

— Flags

— Command code

— Mandatory and optional parameters fields, depending on the command

— Application data fields

— CRC

d) each response consists of the following fields:

— Flags

— Mandatory and optional parameters fields, depending on the command

— Application data fields

— CRC

e) the protocol is bit-oriented. The number of bits transmitted in a frame is a multiple of eight (8), i.e. an integer number of bytes.

f) a single-byte field is transmitted least significant bit (LSBit) first.

g) a multiple-byte field is transmitted least significant byte (LSByte) first, each byte is transmitted least significant bit (LSB) first.

h) the setting of the flags indicates the presence of the optional fields. When the flag is set (to one), the field is present. When the flag is reset (to zero), the field is absent.

i) RFU flags shall be set to zero (0).

7.2 Modes

The term mode refers to the mechanism to specify in a request the set of VICC's that shall answer to the request.

7.2.1 Addressed mode

When the Address_flag is set to 1 (addressed mode), the request shall contain the unique ID (UID) of the addressed VICC.

Any VICC receiving a request with the Address_flag set to 1 shall compare the received unique ID (address) to its own ID.

If it matches, it shall execute it (if possible) and return a response to the VCD as specified by the command description.

If it does not match, it shall remain silent.

7.2.2 Non-addressed mode

When the Address_flag is set to 0 (non-addressed mode), the request shall not contain a unique ID.

Any VICC receiving a request with the Address_flag set to 0 shall execute it (if possible) and shall return a response to the VCD as specified by the command description.

7.2.3 Select mode

When the Select_flag is set to 1 (select mode), the request shall not contain a VICC unique ID.

The VICC in the selected state receiving a request with the Select_flag set to 1 shall execute it (if possible) and shall return a response to the VCD as specified by the command description.

Only the VICC in the selected state shall answer to a request having the select flag set to 1.

7.3 Request format

The request consists of the following fields:

— Flags

— Command code (see clause 10)

— Parameters and data fields

— CRC (see 4.4)

SOF	Flags	Command code	Parameters	Data	CRC	EOF

Figure 4 — General request format

7.3.1 Request flags

In a request, the field "flags" specifies the actions to be performed by the VICC and whether corresponding fields are present or not.

It consists of eight bits.

Table 3 — Request flags 1 to 4 definition

Bit	Flag name	Value	Description
b1	Sub-carrier_flag	0	A single sub-carrier frequency shall be used by the VICC
		1	Two sub-carriers shall be used by the VICC
b2	Data_rate_flag	0	Low data rate shall be used
		1	High data rate shall be used
b3	Inventory_flag	0	Flags 5 to 8 meaning is according to table 4
		1	Flags 5 to 8 meaning is according to table 5
b4	Protocol Extension_flag	0	No protocol format extension
		1	Protocol format is extended. Reserved for future use

NOTE 1 Sub-carrier_flag refers to the VICC-to-VCD communication as specified in ISO/IEC 15693-2.

NOTE 2 Data_rate_flag refers to the VICC-to-VCD communication as specified in ISO/IEC 15693-2.

Table 4 — Request flags 5 to 8 definition when inventory flag is not set

Bit	Flag name	Value	Description
b5	Select_flag	0	Request shall be executed by any VICC according to the setting of Address_flag
		1	Request shall be executed only by VICC in selected state. The Address_flag shall be set to 0 and the UID field shall not be included in the request.
b6	Address_flag	0	Request is not addressed. UID field is not included. It shall be executed by any VICC.
		1	Request is addressed. UID field is included. It shall be executed only by the VICC whose UID matches the UID specified in the request.
b7	Option_flag	0	Meaning is defined by the command description. It shall be set to 0 if not otherwise defined by the command.
		1	Meaning is defined by the command description.
b8	RFU	0	

Table 5 — Request flags 5 to 8 definition when inventory flag is set

Bit	Flag name	Value	Description
b5	AFI_flag	0	AFI field is not present
		1	AFI field is present
b6	Nb_slots_flag	0	16 slots
		1	1 slot
b7	Option_flag	0	Meaning is defined by the command description. It shall be set to 0 if not otherwise defined by the command.
		1	Meaning is defined by the command description.
b8	RFU	0	

7.4 Response format

The response consists of the following fields:

- Flags
- one or more parameter fields
- Data
- CRC (see 4.4)

SOF	Flags	Parameters	Data	CRC	EOF

Figure 5 — General response format

7.4.1 Response flags

In a response, it indicates how actions have been performed by the VICC and whether corresponding fields are present or not.

It consists of eight bits.

Table 6 — Response flags 1 to 8 definition

Bit	Flag name	Value	Description
b1	Error_flag	0	No error
		1	Error detected. Error code is in the "Error" field.
b2	RFU	0	
b3	RFU	0	
b4	Extension_flag	0	No protocol format extension.
		1	Protocol format is extended. Reserved for future use.
b5	RFU	0	
b6	RFU	0	
b7	RFU	0	
b8	RFU	0	

7.4.2 Response error code

When the Error_flag is set by the VICC, the error code field shall be included and provides information about the error that occurred. Error codes are defined in table 7.

If the VICC does not support specific error code(s) listed in table 7, it shall answer with the error code '0F' ("Error with no information given").

Table 7 — Response error code definition

Error code	Meaning
'01'	The command is not supported, i.e. the request code is not recognized.
'02'	The command is not recognized, for example: a format error occurred.
'03'	The command option is not supported.
'0F'	Error with no information given or a specific error code is not supported.
'10'	The specified block is not available (doesn’t exist).
'11'	The specified block is already locked and thus cannot be locked again.
'12'	The specified block is locked and its content cannot be changed.
'13'	The specified block was not successfully programmed.
'14'	The specified block was not successfully locked.
'A0' – 'DF'	Custom command error codes.
all others	RFU

7.5 VICC states

A VICC can be in one of the 4 following states:

— Power-off

— Ready

— Quiet

— Selected

The transition between these states is specified in figure 6.

The support of power-off, ready and quiet states is mandatory.

The support of selected state is optional.

7.5.1 Power-off state

The VICC is in the power-off state when it cannot be activated by the VCD.

7.5.2 Ready state

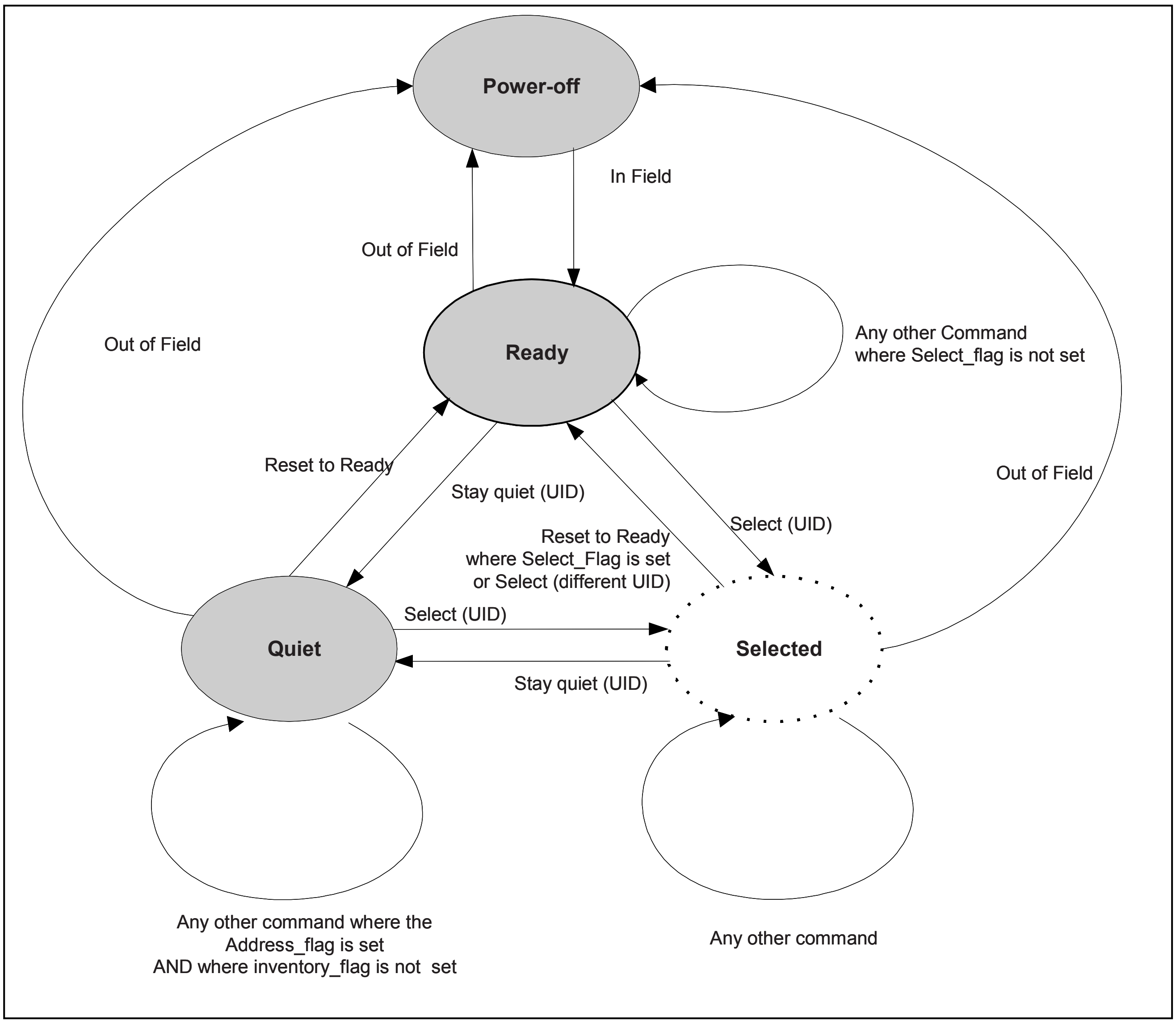
The VICC is in the Ready state when it is activated by the VCD. It shall process any request where the select flag is not set.

7.5.3 Quiet state

When in the quiet state, the VICC shall process any request where the Inventory_flag is not set and where the Address_flag is set.

7.5.4 Selected state

Only a VICC in the selected state shall process requests having the Select_flag set.


Power-off
In Field
Out of Field
Ready
Any other Command where Select_flag is not set
Out of Field
Reset to Ready
Stay quiet (UID)
Out of Field
Select (UID)
Reset to Ready where Select_Flag is set or Select (different UID)
Select (UID)
Quiet
Selected
Stay quiet (UID)
Any other command where the Address_flag is set AND where inventory_flag is not set
Any other command

NOTE 1 The intention of the state transition method is that only one VICC should be in the Selected state at a time.

NOTE 2 The VICC state transition diagram shows only valid transitions. In all other cases the current VICC state remains unchanged. When the VICC cannot process a VCD request (e.g. CRC error, etc.), it shall stay in its current state.

NOTE 3 The Selected state is represented with a dotted line to show its support by the VICC is optional.

Figure 6 — VICC state transition diagram

8 Anticollision

The purpose of the anticollision sequence is to make an inventory of the VICCs present in the VCD field by their unique ID (UID).

The VCD is the master of the communication with one or multiple VICCs. It initiates card communication by issuing the inventory request.

The VICC shall send its response in the slot determined or shall not respond, according to the algorithm described in clause 8.2.

8.1 Request parameters

When issuing the inventory command, the VCD shall set the Nb_slots_flag to the desired setting and add after the command field the mask length and the mask value.

The mask length indicates the number of significant bits of the mask value. It can have any value between 0 and 60 when 16 slots are used and any value between 0 and 64 when 1 slot is used. LSB shall be transmitted first.

The mask value is contained in an integer number of bytes. LSB shall be transmitted first.

If the mask length is not a multiple of 8 (bits), the mask value MSB shall be padded with the required number of null (set to 0) bits so that the mask value is contained in an integer number of bytes.

The next field starts on the next byte boundary.

SOF	Flags	Command	Mask length	Mask value	CRC16	EOF
	8 bits	8 bits	8 bits	0 to 8 bytes	16 bits	

Figure 7 — Inventory request format

MSB	LSB
0000	0100 1100 1111
Pad	Mask value

Figure 8 — Example of the padding of the mask

In the example of figure 8, the mask length is 12 bits. The mask value MSB is padded with four bits set to 0.

The AFI field shall be present if the AFI_flag is set.

The pulse shall be generated according to the definition of the EOF in ISO/IEC 15693-2.

The first slot starts immediately after the reception of the request EOF.

To switch to the next slot, the VCD sends an EOF. The rules, restrictions and timing are specified in clause 9.

8.2 Request processing by the VICC

Upon reception of a valid request, the VICC shall process it by executing the operation sequence specified in the following text in italics. The step sequence is also graphically represented in figure 9.

NbS is the total number of slots (1 or 16)

SN is the current slot number (0 to 15)

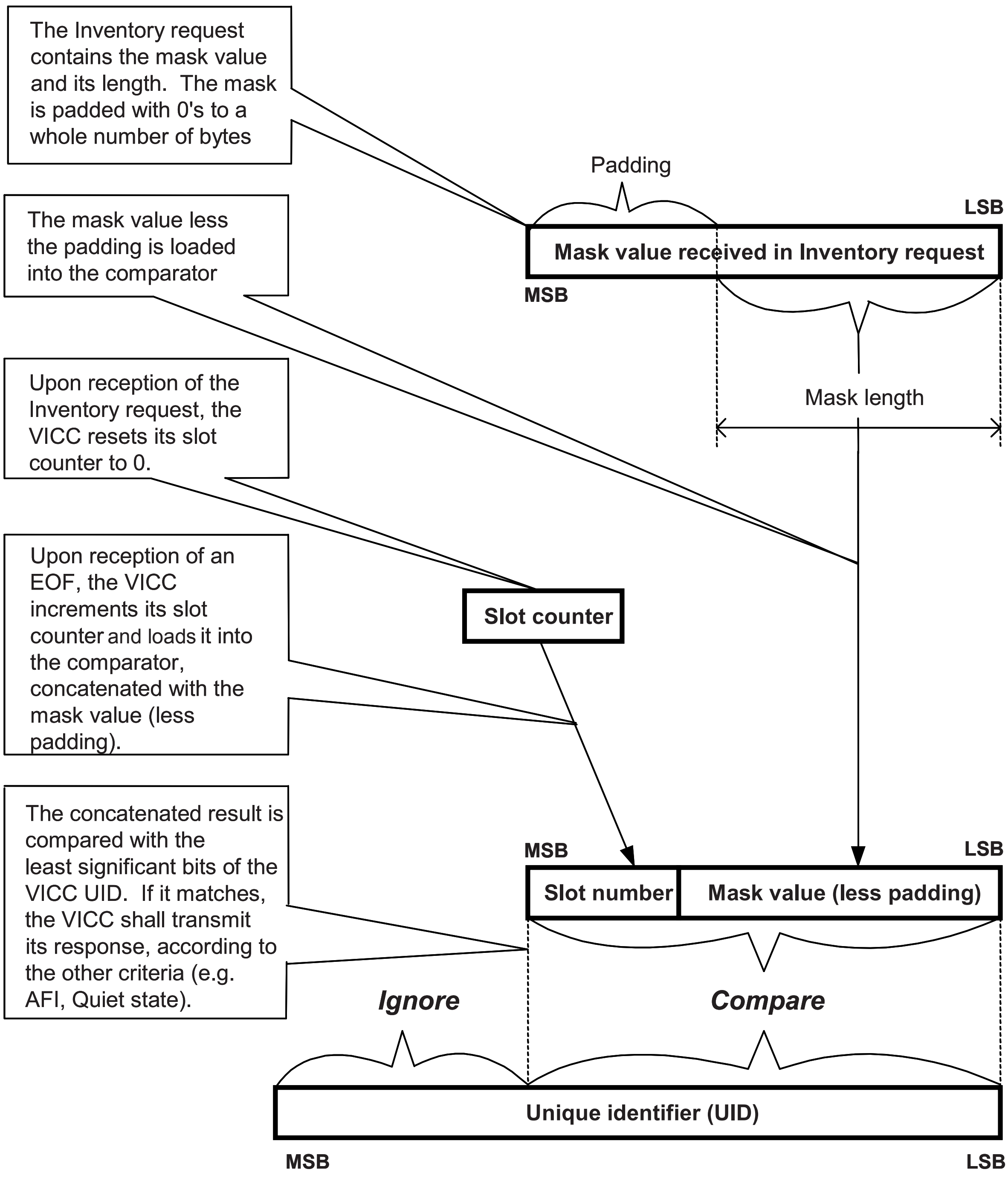
SN_length is set to 0 when 1 slot is used and set to 4 when 16 slots are used

LSB (value, n) function returns the n less significant bits of value

"&" is the concatenation operator

Slot_Frame is either a SOF or an EOF

```
    SN= 0
    if Nb_slots_flag then
        NbS =1 SN_length=0
        else NbS = 16 SN_length=4
    endif
label1: if LSB(UID, SN_length + Mask_length) = LSB(SN, SN_length)&LSB(Mask, Mask_length) then
        transmit response to inventory request
    endif
    wait (Slot_Frame)
    if Slot_Frame= SOF then
        Stop anticollision and decode/process request
        exit
    endif
    if SN<NbS-1 then
        SN = SN +1
        goto label1
        exit
    endif
    exit
```


The Inventory request contains the mask value and its length. The mask is padded with 0's to a whole number of bytes
Padding
LSB
Mask value received in Inventory request
MSB
The mask value less the padding is loaded into the comparator
Mask length
Upon reception of the Inventory request, the VICC resets its slot counter to 0.
Upon reception of an EOF, the VICC increments its slot counter and loads it into the comparator, concatenated with the mask value (less padding).
Slot counter
The concatenated result is compared with the least significant bits of the VICC UID. If it matches, the VICC shall transmit its response, according to the other criteria (e.g. AFI, Quiet state).
MSB
LSB
Slot number
Mask value (less padding)
Ignore
Compare
Unique identifier (UID)
MSB
LSB

NOTE When the slot number is 1 (Nb_slots_flag is set to 1), the comparison is made only on the mask (without padding).

Figure 9 — Principle of comparison between the mask value, slot number and UID

8.3 Explanation of an anticollision sequence

Figure 10 summarises the main cases that can occur during a typical anticollision sequence where the number of slots is 16.

The different steps are:

a) the VCD sends an inventory request, in a frame, terminated by a EOF. The number of slots is 16.

b) VICC 1 transmits its response in slot 0. It is the only one to do so, therefore no collision occurs and its UID is received and registered by the VCD;

c) the VCD sends an EOF, meaning to switch to the next slot.

d) in slot 1, two VICCs 2 and 3 transmits their response, this generates a collision. The VCD detects it and remembers that a collision was detected in slot 1.

e) the VCD sends an EOF, meaning to switch to the next slot.

f) in slot 2, no VICC transmits a response. Therefore the VCD does not detect a VICC SOF and decides to switch to the next slot by sending a EOF.

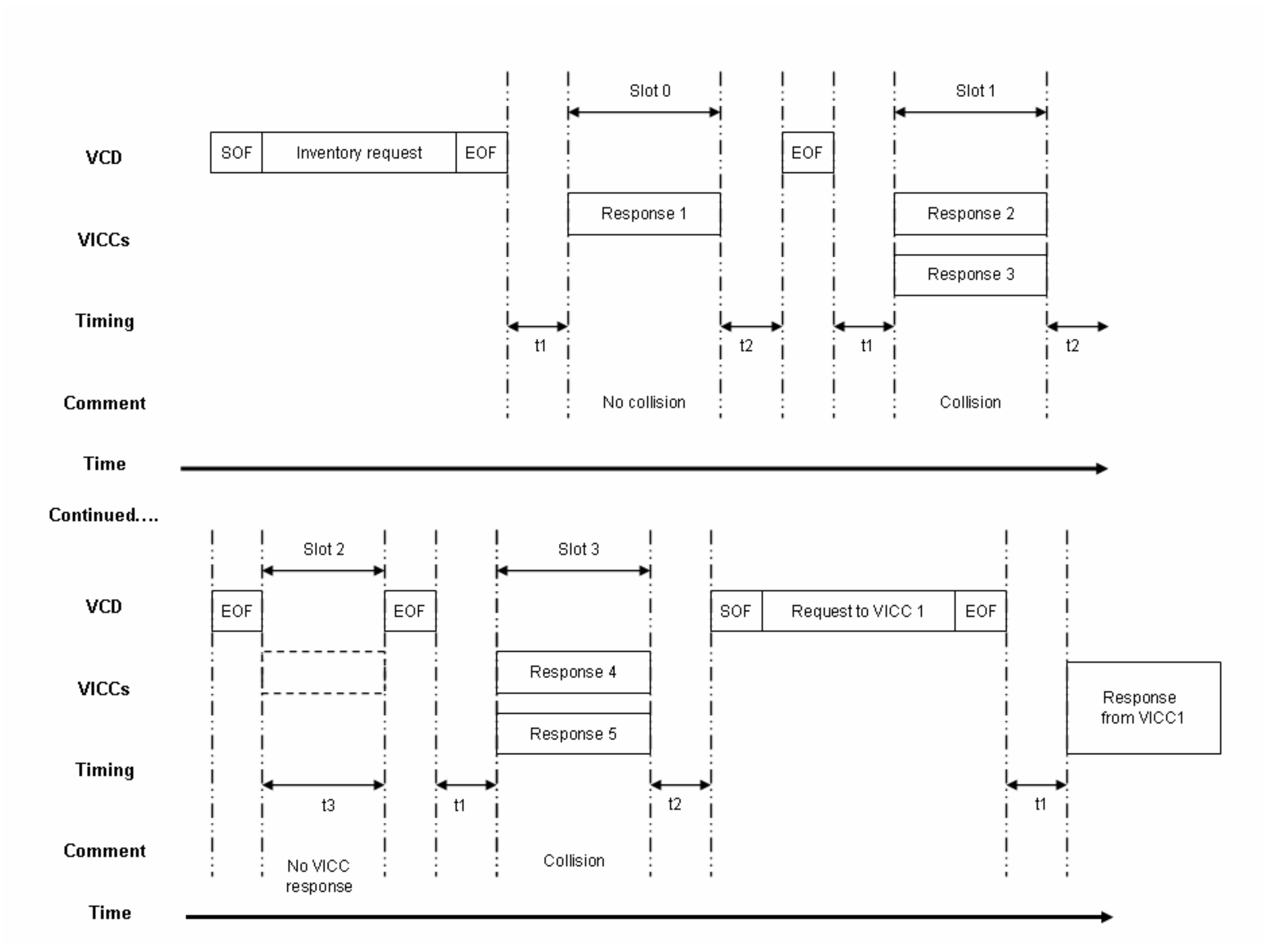
g) in slot 3, there is another collision caused by responses from VICC 4 and 5

h) the VCD then decides to send an addressed request (for instance a Read Block) to VICC 1, which UID was already correctly received.

i) all VICCs detect a SOF and exit the anticollision sequence. They process this request and since the request is addressed to VICC 1, only VICC1 transmit its response.

j) all VICCs are ready to receive another request. If it is an inventory command, the slot numbering sequence restarts from 0.

NOTE The decision to interrupt the anticollision sequence is up to the VCD. It could have continued to send EOF's till slot 15 and then send the request to VICC 1.



NOTE t1, t2 and t3 are specified in clause 9.

Figure 10 — Description of a possible anticollision sequence

9 Timing specifications

The VCD and the VICC shall comply with the following timing specifications.

9.1 VICC waiting time before transmitting its response after reception of an EOF from the VCD

When the VICC has detected an EOF of a valid VCD request or when this EOF is in the normal sequence of a valid VCD request, it shall wait for a time t1 before starting to transmit its response to a VCD request or before switching to the next slot when in an inventory process (see 8.2 and 8.3).

t1 starts from the detection of the rising edge of the EOF received from the VCD (see ISO/IEC 15693-2).

NOTE The synchronisation on the rising edge of the VCD-to-VICC EOF is needed for ensuring the required synchronization of the VICC responses.

The minimum value of t1 is t1min= 4320/f_c (318,6 µs)

The nominal value of t1 is t1nom= 4352/f_c (320,9 µs)

The maximum value of t1 is t1max= 4384/f_c (323,3 µs)

t1max does not apply for Write alike requests. Timing conditions for Write alike requests are defined in the command descriptions.

If the VICC detects a carrier modulation during this time t1, it shall reset its t1 timer and wait for a further time t1 before starting to transmit its response to a VCD request or to switch to the next slot when in an inventory process.

9.2 VICC modulation ignore time after reception of an EOF from the VCD

When the VICC has detected an EOF of a valid VCD request or when this EOF is in the normal sequence of a valid VCD request, it shall ignore any received 10 % modulation during a time $t_{mit.}$

t_{mit} starts from the detection of the rising edge EOF received from the VCD (see ISO/IEC 15693-2).

The minimum value of t_{mit} is t_{mit}min = 4384/f_c (323,3 µs) + t_{nrt}

Where

— t_{nrt} is the nominal response time of a VICC.

t_{nrt} is dependent on the VICC-to-VCD data rate and subcarrier modulation mode (see ISO/IEC 15693-2).

NOTE The synchronisation on the rising edge of the VCD-to-VICC EOF is needed for ensuring the required synchronization of the VICC responses.

9.3 VCD waiting time before sending a subsequent request

a) When the VCD has received a VICC response to a previous request other than Inventory and Quiet, it shall wait a time t2 before sending a subsequent request. t2 starts from the time the EOF has been received from the VICC.

b) When the VCD has sent a Quiet request (which causes no VICC response), it shall wait a time t2 before sending a subsequent request. t2 starts from the end of the Quiet request EOF (rising edge of the EOF plus 9,44 µs, see ISO/IEC 15693-2).

The minimum value of t2 is t2min = 4192/f_c (309,2 µs).

NOTE 1 This ensures that the VICCs are ready to receive this subsequent request (see ISO/IEC 15693-2).

NOTE 2 The VCD should wait at least 1 ms after it activated the powering field before sending the first request, to ensure that the VICCs are ready to receive it (see ISO/IEC 15693-2).

c) When the VCD has sent an Inventory request, it is in an inventory process. See 9.4.

9.4 VCD waiting time before switching to the next slot during an inventory process

An inventory process is started when the VCD sends an Inventory request. (See 8.2, 8.3, 10.3.1),

To switch to the next slot, the VCD may send either a 10 % or a 100 % modulated EOF independent of the modulation index it used for transmitting its request to the VICC, after waiting a time specified in 9.3.a and 9.3.b.

9.4.1 When the VCD has started to receive one or more VICC responses

During an inventory process, when the VCD has started to receive one or more VICC responses (i.e. it has detected a VICC SOF and/or a collision), it shall

— wait for the complete reception of the VICC responses (i.e. when a VICC EOF has been received or when the VICC nominal response time t_{nrt} has elapsed),

— wait an additional time t2

— and then send a 10 % or 100 % modulated EOF to switch to the next slot.

t2 starts from the time the EOF has been received from the VICC (see ISO/IEC 15693-2).

The minimum value of t2 is t2min = 4192/f_c (309,2 µs).

t_{nrt} is dependent on the VICC-to-VCD data rate and subcarrier modulation mode (see ISO/IEC 15693-2).

9.4.2 When the VCD has received no VICC response

During an inventory process, when the VCD has received no VICC response, it shall wait a time t3 before sending a subsequent EOF to switch to the next slot.

t3 starts from the time the VCD has generated the rising edge of the last sent EOF.

a) the VCD sends a 100 % modulated EOF,

the minimum value of t3 is t3min = 4384/f_c (323,3 µs) + t_{sof}

b) If the VCD sends a 10 % modulated EOF,

the minimum value of t3 is t3min = 4384/f_c (323,3 µs) + tnrt + t2min

where

— t_{sof} is the time duration for a VICC to transmit an SOF to the VCD.

— t_{nrt} is the nominal response time of a VICC.

t_{nrt} and t_{sof} are dependent on the VICC-to-VCD data rate and subcarrier modulation mode (see ISO/IEC 15693-2).

A2 9.5 Clarification of use of Option Flag in programming command

If the Option_flag is not set

The VICC shall return its response when it has completed the operation starting after:

t1nom [4352/*fc* (320,9 µs), see 9.1.1] + a multiple of 4096/*fc* (302 µs) with a total tolerance of ± 32/*fc* and latest after 20 ms upon detection of the rising edge of the EOF of the VCD request.

If the Option_flag is set

The VICC shall wait for the reception of an EOF from the VCD and upon such reception shall return its response.

The VCD shall transmit an EOF at least 10 ms and no later than 20 ms after transmitting the command.

NOTE If the VCD transmits an EOF within 10 ms, the VICC may not be able to execute the command or it may reset.

Upon reception of an EOF the VICC shall wait for a duration of t1 before transmitting its response (see 9.1.1) A2

10 Commands

10.1 Command types

Four sets of commands are defined: mandatory, optional, custom, proprietary.

All VICCs with the same IC manufacturer code and same IC version number shall behave the same.

10.1.1 Mandatory

The command codes range from '01' to '1F'.

All VICCs shall support them.

10.1.2 Optional

The command codes range from '20' to '9F'.

VICCs may support them, at their option. If supported, request and response formats shall comply with the definition given in this part of ISO/IEC 15693.

If the VICC does not support an optional command and if the Address_flag or the Select_flag is set, it may return an error code ("Not supported") or remain silent. If neither the Address_flag nor the Select_flag is set, the VICC shall remain silent.

If a command has different options they may be supported by the VICC otherwise an error code shall be returned.

10.1.3 Custom

The command codes range from 'A0' to 'DF'.

VICCs support them, at their option, to implement manufacturer specific functions. The function of flags (including reserved bits) shall not be modified except the Option_flag. The only fields that can be customized are the parameters and the data fields.

Any custom command contains as its first parameter the IC manufacturer code. This allows IC manufacturers to implement custom commands without risking duplication of command codes and thus misinterpretation.

If the VICC does not support a custom command and if the Address_flag or the Select_flag is set, it may return an error code ("Not supported") or remain silent. If neither the Address_flag nor the Select_flag is set, the VICC shall remain silent.

If a command has different options they may be supported by the VICC otherwise an error code shall be returned.

10.1.4 Proprietary

The command codes range from 'E0' to 'FF'.

These commands are used by IC and VICC manufacturers for various purposes such as tests, programming of system information, etc. They are not specified in this part of ISO/IEC 15693. The IC manufacturer may at its option document them or not. It is allowed that these commands are disabled after IC and/or VICC manufacturing.

10.2 Command codes

Table 8 — Command codes

A3⟩

Command code	Type	Function
‘01’	Mandatory	Inventory
‘02’	Mandatory	Stay quiet
‘03’ – ‘1F’	Mandatory	RFU
‘20’	Optional	Read single block
‘21’	Optional	Write single block
‘22’	Optional	Lock block
‘23’	Optional	Read multiple blocks
‘24’	Optional	Write multiple blocks
‘25’	Optional	Select
‘26’	Optional	Reset to ready
‘27’	Optional	Write AFI
‘28’	Optional	Lock AFI
‘29’	Optional	Write DSFID
‘2A’	Optional	Lock DSFID
‘2B’	Optional	Get system information
‘2C’	Optional	Get multiple block security status
‘30’	Optional	Extended read single block
‘31’	Optional	Extended write single block
‘32’	Optional	Extended lock block
‘33’	Optional	Extended read multiple blocks
‘34’	Optional	Extended write multiple blocks
‘3C’	Optional	Extended get multiple block security status
‘2D’ – ‘2F’ ‘35’ – ‘3B’ ‘3D’ – ‘9F’	Optional	RFU
‘A0’ – ‘DF’	Custom	IC Mfg dependent
‘E0’ – ‘FF’	Proprietary	IC Mfg dependent

⟨A3

10.3 Mandatory commands

10.3.1 Inventory

Command code = '01'

When receiving the Inventory request, the VICC shall perform the anticollision sequence.

The request contains:

- The flags,
- The Inventory command code
- The AFI if the AFI flag is set
- The mask length
- The mask value
- The CRC

The Inventory_flag shall be set to 1.

The meaning of flags 5 to 8 is according to table 5.

SOF	Flags	Inventory	Optional AFI	Mask length	Mask value	CRC16	EOF
	8 bits	8 bits	8 bits	8 bits	0 - 64 bits	16 bits	

Figure 11 — Inventory request format

The response shall contain:

- The DSFID
- The unique ID

If the VICC detects an error, it shall remain silent.

SOF	Flags	DSFID	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 12 — Inventory response format

10.3.2 Stay quiet

Command code = '02'

When receiving the Stay quiet command, the VICC shall enter the quiet state and shall NOT send back a response. There is NO response to the Stay quiet command

When in quiet state:

- the VICC shall not process any request where Inventory_flag is set,
- the VICC shall process any addressed request

The VICC shall exit the quiet state when:

— reset (power off),

— receiving a Select request. It shall then go to the selected state if supported or return an error if not supported,

— receiving a Reset to ready request. It shall then go to the Ready state.

SOF	Flags	Stay quiet	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 13 — Stay quiet request format

Request parameter:

UID (mandatory)

The Stay quiet command shall always be executed in Addressed mode (Select_flag is set to 0 and Address_flag is set to 1).

10.4 Optional commands

10.4.1 Read single block

Command code = '20'

When receiving the Read single block command, the VICC shall read the requested block and send back its value in the response.

If the Option_flag is set in the request, the VICC shall return the block security status, followed by the block value.

If it is not set, the VICC shall return only the block value.

SOF	Flags	Read single block	**UID**	Block number	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	16 bits	

Figure 14 — Read single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 15 — Read single block response format when Error_flag is set

SOF	Flags	**Block security status**	Data	CRC16	EOF
	8 bits	**8 bits**	Block length	16 bits	

Figure 16 — Read single block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set

Block security status (if Option_flag is set in the request)

Block data

10.4.2 Write single block

Command code = '21'

When receiving the Write single block command, the VICC shall write the requested block with the data contained in the request and report the success of the operation in the response.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Write single block	**UID**	Block number	Data	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	Block length	16 bits	

Figure 17 — Write single block request format

Request parameter:

(Optional) UID

Block number

Data

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 18 — Write single block response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 19 — Write single block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.3 Lock block

Command code = '22'

When receiving the Lock block command, the VICC shall lock permanently the requested block.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Lock block	**UID**	Block number	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	16 bits	

Figure 20 — Lock single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 21 — Lock block response format Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 22 — Lock block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.4 Read multiple blocks

Command code = '23'

When receiving the Read multiple blocks command, the VICC shall read the requested block(s) and send back their value in the response.

If the Option_flag is set in the request, the VICC shall return the block security status, followed by the block value sequentially block by block.

If the Option_flag is not set in the request, the VICC shall return only the block value.

The blocks are numbered from '00' to 'FF' (0 to 255).

The number of blocks in the request is one less than the number of blocks that the VICC shall return in its response.

EXAMPLE A value of '06' in the "Number of blocks" field requests to read 7 blocks. A value of '00' requests to read a single block.

SOF	Flags	Read multiple block	**UID**	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	8 bits	16 bits	

Figure 23 — Read multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 24 — Read multiple blocks response format when Error_flag is set

SOF	Flags	Block security status	Data	CRC16	EOF
	8 bits	8 bits	Block length	16 bits	
		Repeated as needed			

Figure 25 — Read multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set (the following order shall be respected in the VICC response)

Block security status N (if Option_flag is set in the request)

Block value N

Block security status N+1 (if Option_flag is set in the request)

Block value N+1

etc.

where N is the first requested (and returned) block.

10.4.5 Write multiple blocks

Command code = '24'

When receiving the Write multiple blocks command, the VICC shall write the requested block(s) with the data contained in the request and report the success of the operation in the response.

A2 Option_flag definition see 9.5. A2

The blocks are numbered from '00' to 'FF' (0 to 255).

The number of blocks in the request is one less than the number of blocks that the VICC shall write.

EXAMPLE A value of '06' in the "Number of blocks" field requests to write 7 blocks. The "Data" field shall contain 7 blocks. A value of '00' in the "Number of blocks" field requests to write 1 block. The "Data" field shall contain 1 block.

SOF	Flags	Write multiple block	**UID**	First block number	Number of blocks	Data	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	8 bits	Block length	16 bits	
						Repeated as needed		

Figure 26 — Write multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

Block data (repeated as defined in figure 26)

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 27 — Write multiple blocks response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 28 — Write multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.6 Select

Command code = '25'

When receiving the Select command:

— if the UID is equal to its own UID, the VICC shall enter the selected state and shall send a response.

— if the UID is different to its own and in selected state, the VICC shall return to the Ready state and shall not send a response. The Select command shall always be executed in Addressed mode. (The Select_flag is set to 0. The Address_flag is set to 1.)

— if the UID is different to its own and not in selected state, the VICC shall remain in its state and shall not send a response.

SOF	Flags	Select	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 29 — Select request format

Request parameter:

UID (mandatory)

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 30 — Select response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 31 — Select response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.7 Reset to ready

Command code = '26'

When receiving a Reset to ready command, the VICC shall return to the Ready state.

SOF	Flags	Reset to ready	**UID**	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	

Figure 32 — Reset to ready request format

Request parameter:

(Optional) UID

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 33 — Reset to ready response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 34 — Reset to ready response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.8 Write AFI

Command code = '27'

When receiving the Write AFI request, the VICC shall write the AFI value into its memory.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Write AFI	**UID**	AFI	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	16 bits	

Figure 35 — Write AFI request format

Request parameter:

(Optional) UID

AFI

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 36 — Write AFI response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 37 — Write AFI response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.9 Lock AFI

Command code = '28'

When receiving the Lock AFI request, the VICC shall lock the AFI value permanently into its memory.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Lock AFI	**UID**	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	

Figure 38 — Lock AFI request format

Request parameter:

(Optional) UID

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 39 — Lock AFI response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 40 — Lock AFI response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.10 Write DSFID command

Command code = '29'

When receiving the Write DSFID request, the VICC shall write the DSFID value into its memory.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Write DSFID	**UID**	DSFID	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	16 bits	

Figure 41 — Write DSFID request format

Request parameter:

(Optional) UID

DSFID

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 42 — Write DSFID response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 43 — Write DSFID response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.11 Lock DSFID

Command code = '2A'

When receiving the Lock DSFID request, the VICC shall lock the DSFID value permanently into its memory.

A2 Option_flag definition see 9.5. A2

SOF	Flags	Lock DSFID	**UID**	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	

Figure 44 — Lock DSFID request format

Request parameter:

(Optional) UID

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 45 — Lock DSFID response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 46 — Lock DSFID response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.12 Get system information

Command code = '2B'

This command allows for retrieving the system information value from the VICC.

SOF	Flags	Get system info	**UID**	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	

Figure 47 — Get system information request format

Request parameter:

(Optional) UID

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 48 — Get system information response when Error_flag is set

SOF	Flags	Info flags	UID	**DSFID**	**AFI**	**Other fields**	CRC16	EOF
	8 bits	8 bits	64 bits	**8 bits**	**8 bits**	**See below**	16 bits	

Figure 49 — Get system information response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set

Information flag

UID (mandatory)

Information fields, in the order of their corresponding flag, as defined in figure 49 and table 9, if their corresponding flag is set.

Table 9 — Information flags definition

Bit	Flag name	Value	Description
b1	DSFID	0	DSFID is not supported. DSFID field is not present
		1	DSFID is supported. DSFID field is present
b2	AFI	0	AFI is not supported. AFI field is not present
		1	AFI is supported. AFI field is present
b3	VICC memory size	0	Information on VICC memory size is not supported. Memory size field is not present.
		1	Information on VICC memory size is supported. Memory size field is present.
b4	IC reference	0	Information on IC reference is not supported. IC reference field is not present.
		1	Information on IC reference is supported. IC reference field is present.
b5	RFU	0	
b6	RFU	0	
b7	RFU	0	
b8	RFU	0	

Table 10 — VICC memory size information

MSB		LSB
16 14	13 9	8 1
RFU	Block size in bytes	Number of blocks

Block size is expressed in number of bytes on 5 bits, allowing to specify up to 32 bytes i.e. 256 bits. It is one less than the actual number of bytes.

EXAMPLE A value of '1F' indicates 32 bytes, a value of '00' indicates 1 byte.

Number of blocks is on 8 bits, allowing to specify up to 256 blocks. It is one less than the actual number of blocks.

EXAMPLE A value of 'FF' indicates 256 blocks, a value of '00' indicates 1 block.

The three most significant bits are reserved for future use and shall be set to zero.

The IC reference is on 8 bits and its meaning is defined by the IC manufacturer.

10.4.13 Get multiple block security status

Command code = '2C'

When receiving the Get multiple block security status command, the VICC shall send back the block security status.

The blocks are numbered from '00 to 'FF' (0 to 255).

The number of blocks in the request is one less than the number of block security status that the VICC shall return in its response.

EXAMPLE A value of '06' in the "Number of blocks" field requests to return 7 Block security status. A value of '00' in the "Number of blocks" field requests to return a single Block security status.

SOF	Flags	Get multiple block security status	**UID**	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	**64 bits**	8 bits	8 bits	16 bits	

Figure 50 — Get multiple block security status request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 51 — Get multiple block security status response format when Error_flag is set

SOF	Flags	Block security status	CRC16	EOF
	8 bits	8 bits	16 bits	
		Repeated as needed		

Figure 52 — Get multiple block security status response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set Block security status (repeated as per figure 52)

A3 10.4.14 Extended read single block

Command code = '30'

When receiving the Extended read single block command, the VICC shall read the requested block and send back its value in the response.

If a VICC supports Extended read single block command, it shall also support Read single block command for the first 256 blocks of memory.

If the Option_flag is set in the request, the VICC shall return the block security status, followed by the block value.

If it is not set, the VICC shall return only the block value.

SOF	Flags	Extended read single block	**UID**	Block number	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	16 bits	

Figure 53 — Extended read single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 54 — Extended read single block response format when Error_flag is set

SOF	Flags	**Block security status**	Data	CRC16	EOF
	8 bits	**8 bits**	Block length	16 bits	

Figure 55 — Extended read single block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set

Block security status (if Option_flag is set in the request)

Block data A3

A3 10.4.15 Extended write single block

Command code = '31'

When receiving the Extended write single block command, the VICC shall write the requested block with the data contained in the request and report the success of the operation in the response.

If a VICC supports Extended write single block command, it shall also support Write single block command for the first 256 blocks of memory.

Option_flag definition, see ISO/IEC 15693-3:2009/Amd2:20XX, Clause 9.5.

SOF	Flags	Extended write single block	**UID**	Block number	Data	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	Block length	16 bits	

Figure 56 — Extended write single block request format

Request parameter:

(Optional) UID

Block number

Data

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 57 — Extended write single block response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 58 — Extended write single block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set) A3

A3 10.4.16 Extended lock block

Command code = '32'

When receiving the Extended lock block command, the VICC shall lock permanently the requested block.

If a VICC supports Extended lock block command, it shall also support Lock block command for the first 256 blocks of memory.

Option_flag definition, see ISO/IEC 15693-3:2009/Amd2:20XX, Clause 9.5.

SOF	Flags	Extended lock block	**UID**	Block number	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	16 bits	

Figure 59 — Extended lock single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 60 — Extended lock block response format Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 61 — Extended lock block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

10.4.17 Extended read multiple blocks

Command code = '33'

When receiving the Extended read multiple blocks command, the VICC shall read the requested block(s) and send back their value in the response.

If a VICC supports Extended read multiple blocks command, it shall also support Read multiple blocks command for the first 256 blocks of memory.

If the Option_flag is set in the request, the VICC shall return the block security status, followed by the block value sequentially block by block.

If the Option_flag is not set in the request, the VICC shall return only the block value.

The blocks are numbered from '0000' to 'FFFF' (0 to 65535). A3

A3 The number of blocks in the request is one less than the number of blocks that the VICC shall return in its response.

EXAMPLE A value of '0006' in the "Number of blocks" field requests to read 7 blocks. A value of '0000' requests to read a single block.

SOF	Flags	Extended read multiple block	**UID**	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	16 bits	16 bits	

Figure 62 — Extended read multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 63 — Extended read multiple blocks response format when Error_flag is set

SOF	Flags	Block security status	Data	CRC16	EOF
	8 bits	8 bits	Block length	16 bits	
		Repeated as needed			

Figure 64 — Extended read multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set (the following order shall be respected in the VICC response)

Block security status N (if Option_flag is set in the request)

Block value N

Block security status N+1 (if Option_flag is set in the request)

Block value N+1

etc.

where N is the first requested (and returned) block. A3

10.4.18 Extended write multiple blocks

Command code = ‘34’

When receiving the Extended write multiple blocks command, the VICC shall write the requested block(s) with the data contained in the request and report the success of the operation in the response.

If a VICC supports Extended write multiple blocks command, it shall also support Write multiple blocks command for the first 256 blocks of memory.

Option_flag definition, see ISO/IEC 15693-3:2009/Amd2:20XX, Clause 9.5.

The blocks are numbered from ‘0000’ to ‘FFFF’ (0 to 65535).

The number of blocks in the request is one less than the number of blocks that the VICC shall write.

EXAMPLE A value of ‘0006’ in the “Number of blocks” field requests to write 7 blocks. The “Data” field shall contain 7 blocks. A value of ‘0000’ in the “Number of blocks” field requests to write 1 block. The “Data” field shall contain 1 block.

SOF	Flags	Extended write multiple block	**UID**	First block number	Number of blocks	Data	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	16 bits	Block length	16 bits	
						Repeated as needed		

Figure 65 — Extended write multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

Block data (repeated as defined in Figure 65)

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 66 — Extended write multiple blocks response format when Error_flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 67 — Extended write multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

A3⟩ 10.4.19 Extended get multiple block security status

Command code = ‘3C’

When receiving the Extended get multiple block security status command, the VICC shall send back the block security status.

The blocks are numbered from ‘0000’ to ‘FFFF’ (0 to 65535).

The number of blocks in the request is one less than the number of block security status that the VICC shall return in its response.

EXAMPLE A value of ‘0006’ in the “Number of blocks” field requests to return 7 Block security status. A value of ‘0000’ in the “Number of blocks” field requests to return a single Block security status.

SOF	Flags	Extended get multiple block security status	**UID**	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	**64 bits**	16 bits	16 bits	16 bits	

Figure 68 — Extended get multiple block security status request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 69 — Extended get multiple block security status response format when Error_flag is set

SOF	Flags	Block security status	CRC16	EOF
	8 bits	8 bits	16 bits	
		Repeated as needed		

Figure 70 — Extended get multiple block security status response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set Block security status (repeated as per Figure 70) ⟨A3

10.5 Custom commands

The format of custom command is generic and allows unambiguous attribution of custom command codes to each VICC manufacturer.

The custom command code is the combination of a custom command code and of the VICC manufacturer code.

The custom request parameters definition is the responsibility of the VICC manufacturer.

SOF	Flags	Custom	IC Mfg code	Custom request parameters	CRC16	EOF
	8 bits	8 bits	8 bits	Custom defined	16 bits	

Figure 53 — Custom request format

Request parameter:

IC manufacturer code according to ISO/IEC 7816-6:2004.

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 54 — Custom response format when Error_flag is set

SOF	Flags	Custom response parameters	CRC16	EOF
	8 bits	Custom defined	16 bits	

Figure 55 — Custom response format when Error_flag is NOT set

Response parameter:

Error_flag (and code if Error_flag is set)

if Error_flag is not set

Custom parameters

10.6 Proprietary commands

The format of these commands is not defined in this part of ISO/IEC 15693.

Annex A
(informative)

Compatibility with other card standards

This part of ISO/IEC 15693 does not preclude the addition of other existing card standards on the VICC, such as ISO/IEC 7816 or others listed in the bibliography.

Annex B
(informative)

VCD pseudo-code for anticollision

The following pseudo-code in italics describes how the anticollision could be implemented on the VCD, using recursivity. It does not describe the collision detection mechanism.

Algorithm for 16 slots

```
    function push (mask, address)           ; pushes on private stack
    function pop (mask, address)            ; pops from private stack
    function pulse_next_pause               ; generates a power pulse
    function store(VICC_UID)                ; stores VICC_UID
    function poll_loop (sub_address_size as integer)

; address length must be four (4) bits.
pop (mask, address)
mask = address & mask                ; generates new mask

; send the Request
mode = anticollision
send_Request(Request_cmd, mode, mask length, mask)

for address = 0 to (2^sub_address_size - 1)
     if no_collision_is_detected then ; VICC is inventoried
          store (VICC_UID)

     else                            ; remember a collision was detected
         push(mask,address)
     endif

pulse_next_pause

next sub_address
; if some collisions have been detected and not yet processed,
; the function calls itself recursively to process the last
; stored collision
if stack_not_empty then poll_loop (sub_address_size)

end poll_loop

main_cycle:
mask = null
address = null
push (mask, address)
poll_loop(sub_address_size)
end_main_cycle
```

Annex C
(informative)

Cyclic Redundancy Check (CRC)

C.1 The CRC error detection method

The Cyclic Redundancy Check (CRC) is calculated on all data contained in a message, from the start of the flags through to the end of data. This CRC is used from VCD to VICC and from VICC to VCD.

Table C.1 — CRC Definition

CRC Definition					
CRC type	**Length**	**Polynomial**	**Direction**	**Preset**	**Residue**
ISO/IEC 13239	16 bits	$X^{16} + X^{12} + X^{5} + 1$	Backward	'FFFF'	'F0B8'

To add extra protection against shifting errors, a further transformation is made on the calculated CRC. The value attached to the message for transmission is the one's complement of the calculated CRC. This transformation is included in the example below.

For checking of received messages the 2 CRC bytes are often also included in the re-calculation, for ease of use. In this case, for the expected value for the generated CRC the value of the residue is 'F0B8'.

The following text in italics is an example which illustrates one method in C language of calculating the CRC on a given set of bytes comprising a message.

```
#include <stdio.h>

#define POLYNOMIAL 0x8408               // x^16 + x^12 + x^5 + 1
#define PRESET_VALUE 0xFFFF
#define CHECK_VALUE 0xF0B8
#define NUMBER_OF_BYTES        4        // Example: 4 data bytes
#define CALC_CRC          1
#define CHECK_CRC            0

void main()
{
unsigned int current_crc_value;
unsigned char array_of_databytes[NUMBER_OF_BYTES + 2] = {1, 2, 3, 4, 0x91, 0x39};
int     number_of_databytes = NUMBER_OF_BYTES;
int     calculate_or_check_crc;
int     i, j;
calculate_or_check_crc = CALC_CRC;
// calculate_or_check_crc = CHECK_CRC; // This could be an other example
if (calculate_or_check_crc == CALC_CRC)
{
        number_of_databytes = NUMBER_OF_BYTES;
}
else // check CRC
{
        number_of_databytes = NUMBER_OF_BYTES + 2;
}
```

```
current_crc_value = PRESET_VALUE;
for (i = 0; i < number_of_databytes; i++)
{
        current_crc_value = current_crc_value ^ ((unsigned int)array_of_databytes[i]);
        for (j = 0; j < 8; j++)
        {
              if (current_crc_value & 0x0001)
              {
                    current_crc_value = (current_crc_value >> 1) ^ POLYNOMIAL;
              }
              else
              {
                    current_crc_value = (current_crc_value >> 1);
              }
        }
}
if (calculate_or_check_crc == CALC_CRC)
 {
        current_crc_value = ~current_crc_value;
        printf ("CRC-ISO/IEC 13239 of { 1, 2, 3, 4 } is '3991'\n");
        printf ("Generated CRC is '%04X'\n", current_crc_value);
              printf ("The Least Significant Byte (transmitted first) is: '%02X'\n",
                    current_crc_value & 0xFF);
              printf( "The Most Significant Byte (transmitted second) is: '%02X'\n",
              (current_crc_value >> 8) & 0xFF);
        printf( "Executing this program when CHECK_CRC generates: 'F0B8'\n");
        // current_crc_value is now ready to be appended to the data stream
        // (first LSByte, then MSByte)
}
else // check CRC
{
        if (current_crc_value == CHECK_VALUE)
        {
              printf ("Checked CRC is ok (0x%04X)\n", current_crc_value);
        }
        else
        {
              printf ("Checked CRC is NOT ok (0x%04X)\n", current_crc_value);
        }
}
```

The above C-program generates the following printout when executed.

```
CRC-ISO/IEC 13239 of { 1, 2, 3, 4 } is '3991'
Generated CRC is '3991'
The Least Significant Byte (transmitted first) is: '91'
The Most Significant Byte (transmitted second) is: '39'
Executing this program when CHECK_CRC generates: 'F0B8'
```

C.2 CRC calculation example

This example refers to a Read single block request for reading block '0B'.

The modes selected by the VCD are: single subcarrier, high VICC-to-VCD data rate, addressed.

The UID of the VICC is: 'E0 04 AB 89 67 45 23 01'

The request therefore consists of the following fields:

- the flags: '22'
- the command code: '20'
- the UID: 'E0 04 AB 89 67 45 23 01' where 'E0' is the most significant byte
- the block number: '0B'
- the CRC: 'BAE3' where 'BA' is the most significant byte.

The CRC is calculated on the request fields assembled in a frame according to the transmission rules defined in this part of ISO/IEC 15693:

'22' '20' '01' '23' '45' '67' '89' 'AB' '04' 'E0' '0B'

NOTE 1 The UID is transmitted least significant byte first.

NOTE 2 Table C.2 describes the different steps of the calculation.

The request is then transmitted as follows:

SOF '22' '20' '01' '23' '45' '67' '89' 'AB' '04' 'E0' '0B' 'E3' 'BA' EOF

NOTE 1 The CRC is transmitted least significant byte first.

NOTE 2 Each byte is transmitted least significant bit first.

Table C.2 — Practical example of CRC calculation

Step	Input	Calculated CRC in VCD	Calculated CRC in VICC for check
1	'22'	'F268	'0D97'
2	'20'	'3EC6'	'C139'
3	'01'	'42F5'	'BD0A'
4	'23'	'4381'	'BC7E'
5	'45'	'7013'	'8FEC'
6	'67'	'C5AB'	'3A54'
7	'89'	'F2AD'	'0D52'
8	'AB'	'95BC'	'6A43'
9	'04'	'C92E'	'36D1'
10	'E0'	'DFC3'	'203C'
11	'0B'	'BAE3'	'451C'
12	'E3'		'0F3D'
13	'BA'		'F0B8'

Bibliography

ISO/IEC card standards

[1] ISO/IEC 7811 (all parts), *Identification cards — Recording technique*

[2] ISO/IEC 7812-1:2006, *Identification cards — Identification of issuers — Part 1: Numbering system*

[3] ISO/IEC 7812-2:2007, *Identification cards — Identification of issuers — Part 2: Application and registration procedures*

[4] ISO/IEC 7813:2006, *Information technology — Identification cards — Financial transaction cards*

[5] ISO/IEC 7816-1:1998, *Identification cards — Integrated circuit(s) cards with contacts — Part 1: Physical characteristics*

[6] ISO/IEC 7816-2:2007, *Identification cards — Integrated circuit cards — Part 2: Cards with contacts — Dimensions and location of the contacts*

[7] ISO/IEC 10373-7, *Identification cards — Test methods — Part 7: Vicinity cards*

ICS 35.240.15

Price based on 43 pages